**PROJEKAT iz predmeta “BAZE PODATAKA 3”**

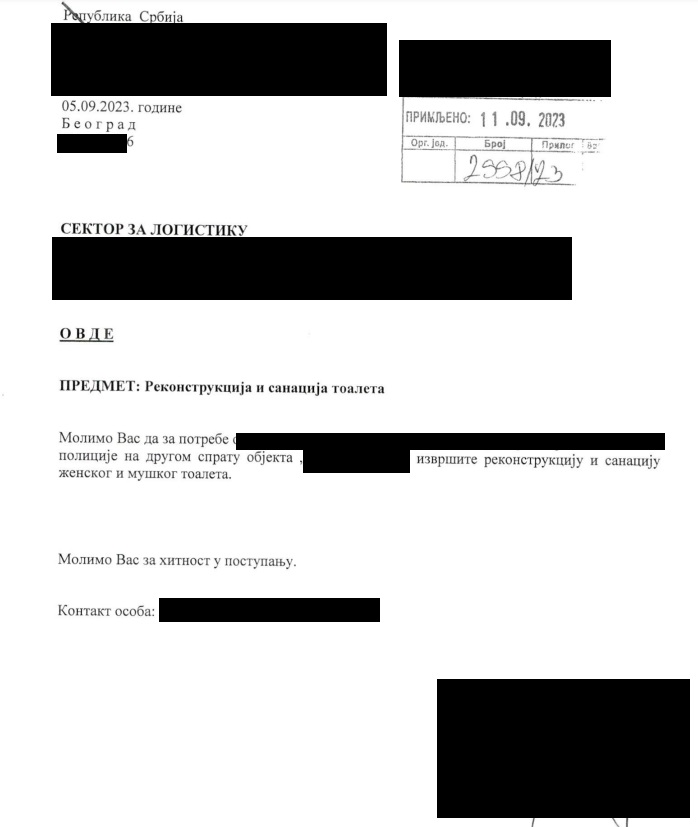
|  |  |  |  |
| --- | --- | --- | --- |
| **IME I PREZIME** | | **Aleksa Đorđević** | |
| Broj indeksa | | **2023/3054** | |
| Studijski modul | | Informacione tehnologije i sajber bezbednost | |
| Stečena diploma osnovnih akademskih studija | | Diplomirani inženjer organizacionih nauka | |
| **NAZIV PROJEKTA** | | Informacioni sistem za proces izvođenja radova na tekućem održavanju | |
| Napomena | |  | |
| *Verzija* | *Datum promene* | | *Opis promene* | |
| 1. | 31.10.2023. | | **Dodati dokumenti I opis** | |
| 2. | 6.11.2023. | | **PMOV i relacioni** | |
| 3. | 13.11.2023. | | **PMOV i relacioni 2. verzija i denormalizacija** | |
| 4. | 24.11.2023. | | **Definisanje korisničkog tipa i implementacija trigera** | |
| 5. | 8.12.2023. | | **Implementacija trigera 2. verzija** | |
| 6. | 10.12.2023 | | **Optimizacija baze** | |

# OPIS PROJEKTA

## Opis podsistema za koji se projektuje baza podataka

|  |
| --- |
| U ovom radu biće projektovan deo sistema koji se odnosi se na proces izvođenja radova na tekućem održavanju. Početak ovog procesa se označava pristizanjem **zahteva za radove**. Nakon što zahtev dobije odobrenje, prvi korak jeste stvaranje **radnog naloga** kao osnovnog dokumenta. Sledeći značajan korak je dodeljivanje ovog zadatka određenim zaposlenima, koji zatim izrađuju detaljnu specifikaciju radova, to jest **predmer radova**. Kada su potrebni radovi precizno identifikovani i dokumentovani u predmeru radova, sledeći korak obuhvata izradu **poziva za dostavu ponude**. Ovaj dokument se koristi za pozivanje potencijalnih izvođača da podnesu svoje ponude za izvođenje radova. Nakon prikupljanja ponuda, proces se nastavlja sa otvaranjem ponuda, što se zvanično beleži putem **zapisnika o otvaranju ponuda**. Finalni korak u ovom procesu obuhvata kreiranje **narudžbenice** sa odabranim izvođačem radova. Ovaj dokument služi kao zvanični ugovor između narudžbodavca i izvođača, definišući uslove i obaveze obe strane. |

## Poslovna dokumenta koja se koriste u procesu



*Slika 1 - Zahtev za izvođenje radova*



*Slika 2 - Radni nalog*

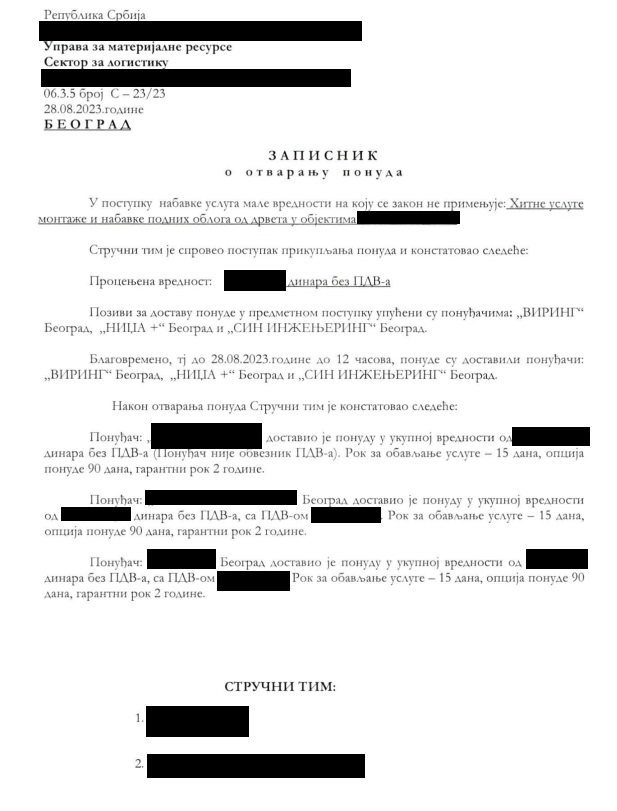
A paper with text on it

Description automatically generated

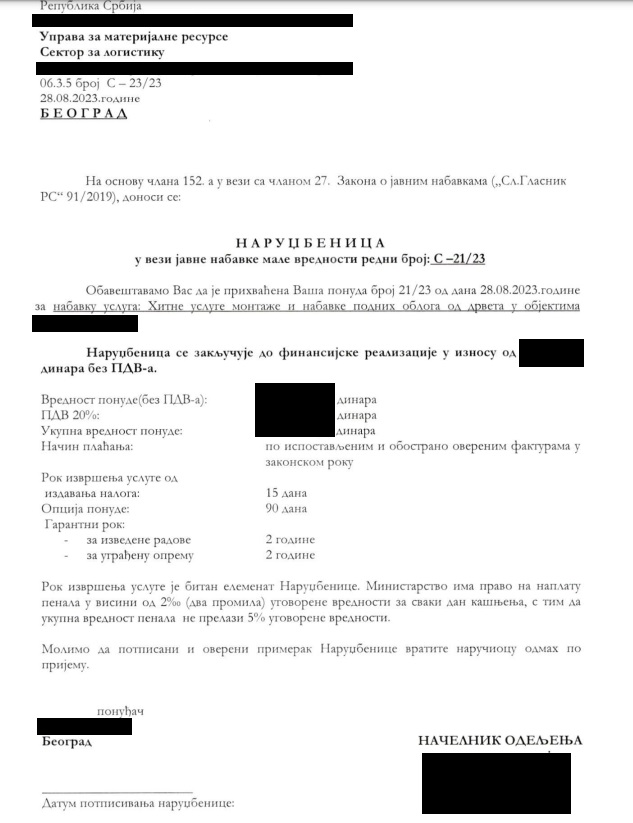
*Slika 3 - Predmer radova*



*Slika 4 - Poziv za dostavu ponude*



*Slika 5 - Zapisnik o otvaranju ponude*

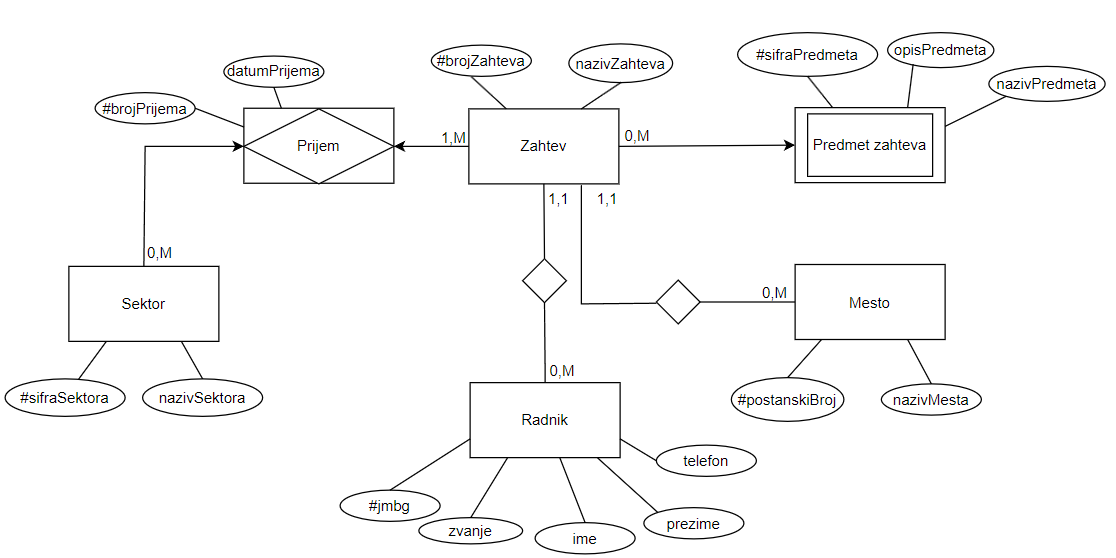


*Slika 6 - Narudžbenica*

# PROJEKTOVANJE BAZE PODATAKA

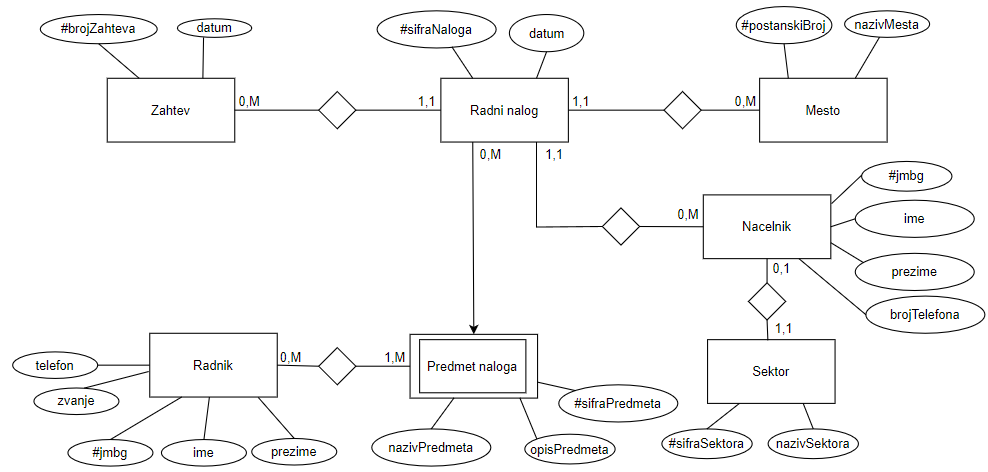
## Model podatka

1. Zahtev za izvođenje radova



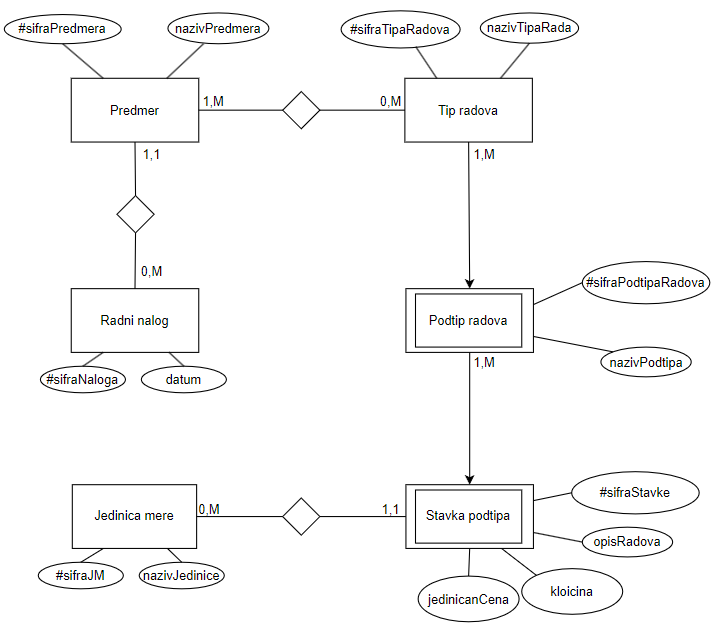
*Slika 7 - PMOV zahtev za izvođenje radova*

1. Radni nalog



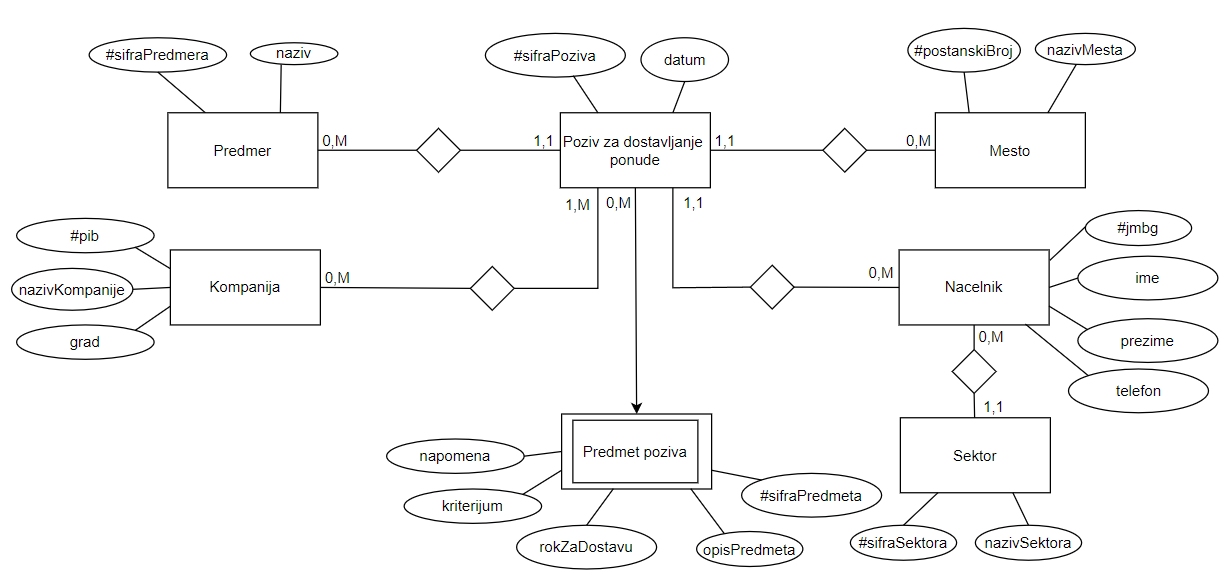
*Slika 8 - PMOV radni nalog*

1. Predmer radova



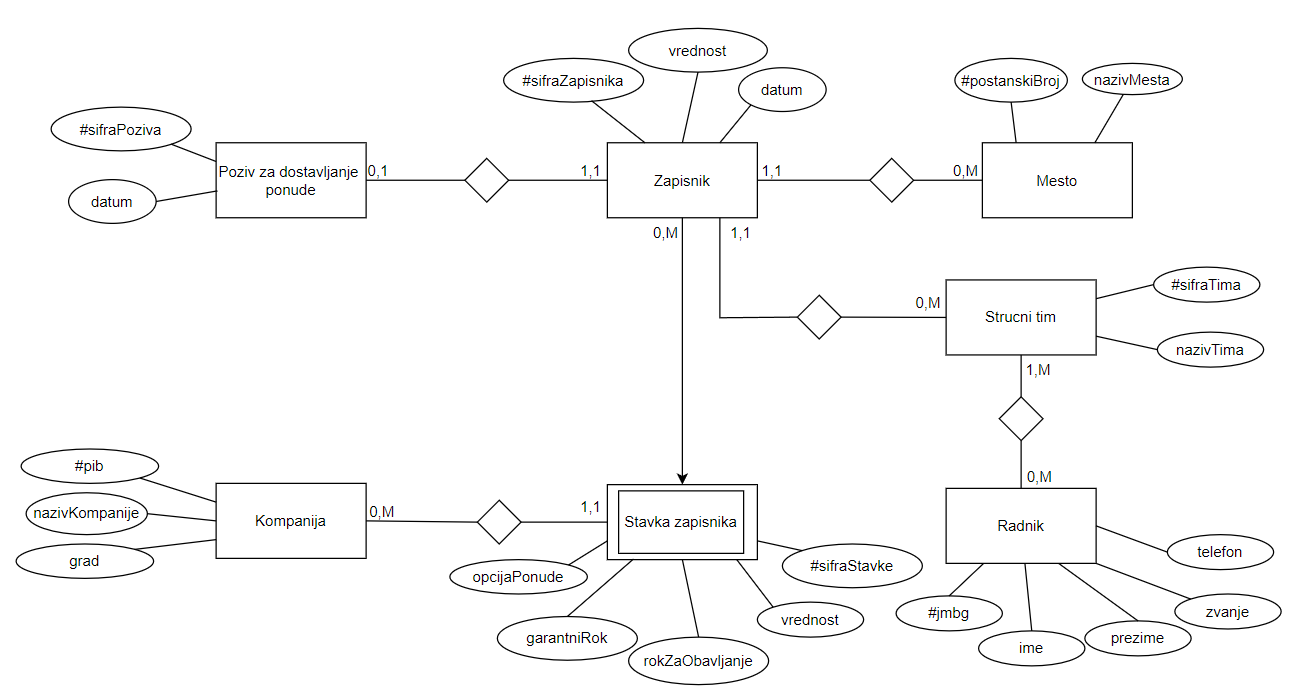
*Slika 9 - PMOV predmer radova*

1. Poziv za dostavu ponude



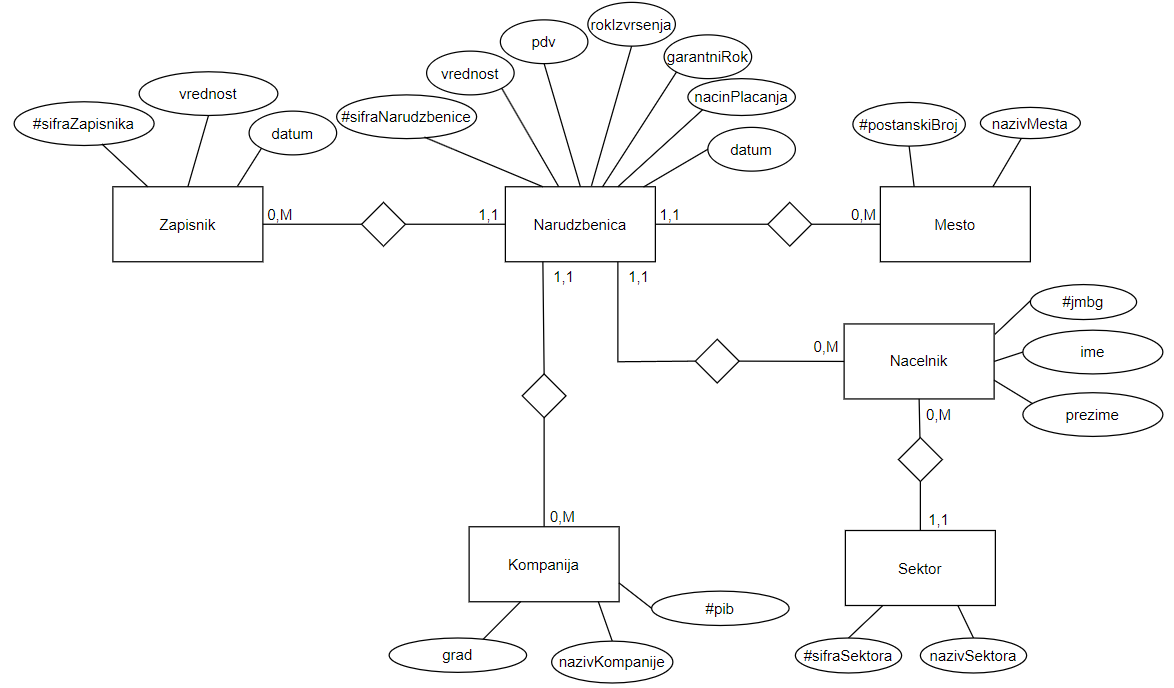
*Slika 10 - PMOV poziv za dostavu ponude*

1. Zapisnik o otvaranju ponude



*Slika 11 - PMOV zapisnik o otvaranju ponude*

1. Narudžbenica



*Slika 12 - PMOV narudžbenica*

## Relacioni model

**Zahtev**(brojZahteva, nazivZahteva, *jmbgR, postanskiBroj*)

PredmetZahteva(*brojZahteva*, sifraPredmeta, opisPredmeta, nazivPredmeta)

Radnik(jmbg, ime, prezime, zvanje, telefon)

Mesto(postanskiBroj, nazivMesta)

Prijem(*brojZahteva*, *sifraSektora*, brojPrijema, datumPrijema)

**RadniNalog**(sifraNaloga, datum, *jmbgN*, *postanskiBroj*, *brojZahteva*)

Nacelnik(jmbg, ime, prezime, brojTelefona)

Sektor(sifraSektora, nazivSektora, *jmbgN*)

PredmetNaloga(*sifraNaloga*, sifraPredmeta, opisPredmeta, nazivPredmeta)

PredmetPoRadniku(*jmbgR*, *sifraNaloga*, sifraPredmeta)

**Predmer**(sifraPredmera, nazivPredmera, *sifraNaloga*)

TipRadova(sifraTipaRadova, nazivTipaRada)

PredmerPoTipu(*sifraPredmera*, *sifraTipaRadova*)

PodtipRadova(*sifraTipaRadova*, sifraPodtipaRadova, nazivPodtipa)

Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCena, *sifraJM*)

JedinicaMere(sifraJM, nazivJedinice)

**PozivZaDostavljanjePonude**(sifraPoziva, datum, *jmbgN*, *postanskiBroj*, *sifraPredmera*)

Kompanija(pib, nazivKompanije, grad)

PozivPoKompaniji(*sifraPoziva*, *pib*)

PredmetPoziva(*sifraPoziva*, sifraPredmeta, napomena, kriterijum, rokZaDostavu, opisPredmeta)

**Zapisnik**(sifraZapisnika, vrednost, datum, *sifraPoziva*, *postanskiBroj*, *siftaTima*)

StavkaZapisnika(sifraZapisnika, *sifraStavke*, vrednost, rokZaObavljanje, garantniRok, opcijaPonude, *pib*)

StrucniTim(sifraTima, nazivTima)

RadnikPoTimu(*SifraTima*, *jmbgR*)

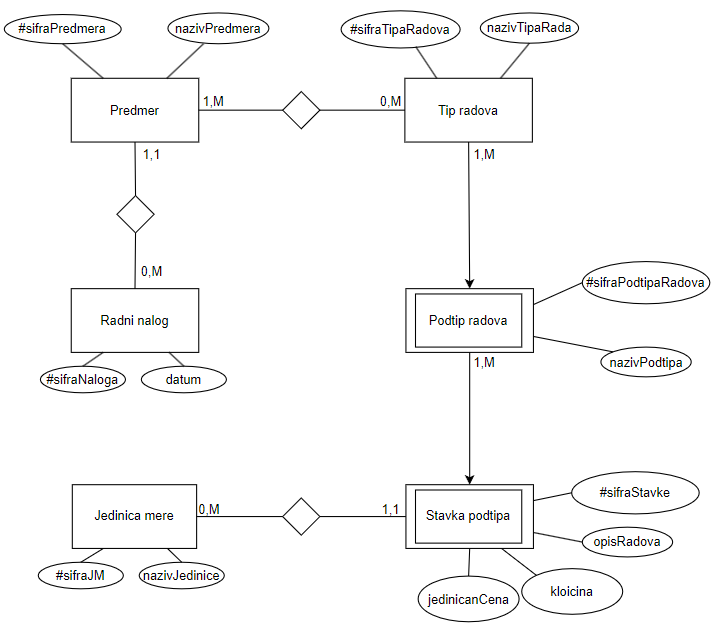
**Narudzbenica**(sifraNarudzbenice, vrednost, pdv, rokIzvrsenja, garantniRok, nacinPlacanja, datum, *postanskiBroj*, *sifraZapisnika*, *jmbgN*, *pib*)

# DENORMALIZACIJA RELACIJA

## Denormalizacija relacija uz narušavanje 2N

Narušavanje 2NF na relaciji **Sredstva** **pre-joining** tehnikom denormalizacije:

**PRE DENORMALIZACIJE**



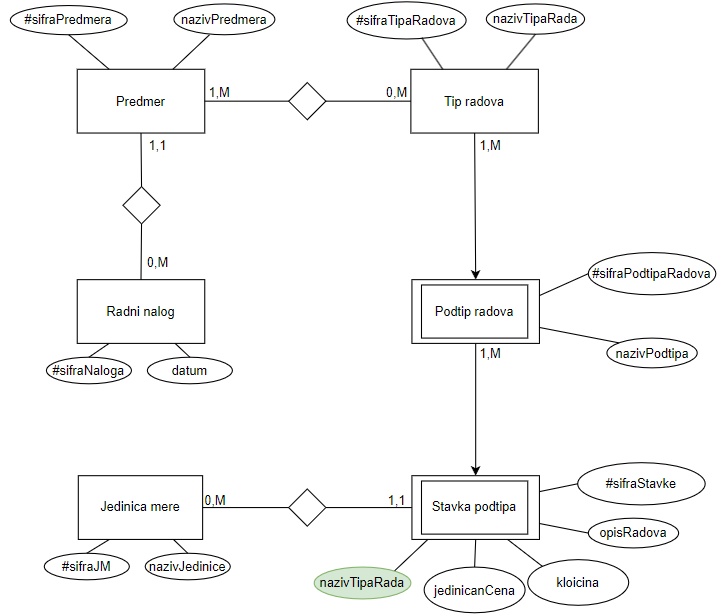
*Slika 13 - Radni nalog pre denormalizacije*

TipRadova(sifraTipaRadova, nazivTipaRada)

Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCena, *sifraJM*)

Funkcionalna zavisnost: sifraTipaRadova, sifraPodtipaRadova, sifraStavke-> opisRadova, kolicina, jedinicnaCena, sifraJM

**NAKON DENORMALIZACIJE**



*Slika 14 - Radni nalog posle denormalizacije*

TipRadova(sifraTipaRadova, nazivTipaRada)

Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCenA, nazivTipaRada, *sifraJM*)

Funkcionalna zavisnost: sifraTipaRadova, sifraPodtipaRadova, sifraStavke-> opisRadova, kolicina, jedinicnaCena, sifraJM, nazivTipaRada

sifraTipaRadova -> nazivTipaRada

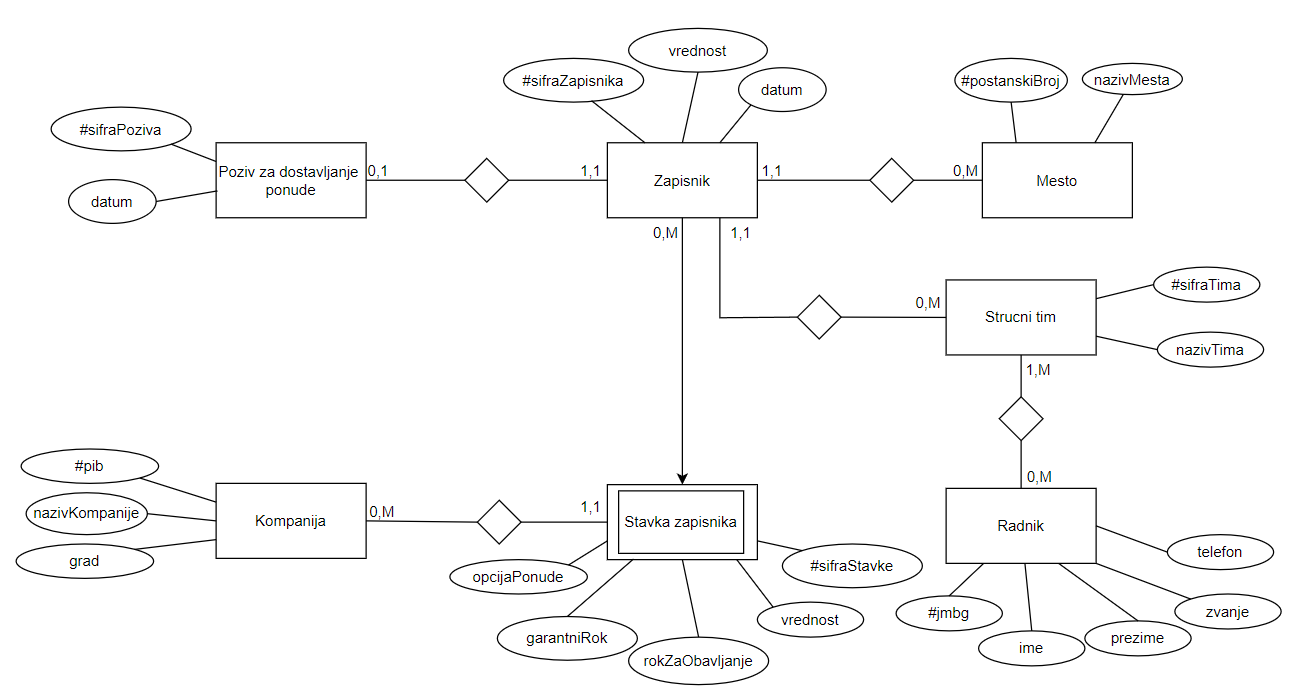
**SPECIFIKACIJA TRIGERA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| TipRadova | Insert |  | Ne |  |
| Update | nazivTipaRada | Da | Nakon izmene vrednosti kolone **nazivTipaRada** u tabeli **TipRadova**, pokreće se triger koji izmenjenu vrednost ažurira i u tabeli **StavkaPodtipa**. |
| Delete |  | Ne |  |
| StavkaPodtipa | Insert |  | Da | U kolonu **nazivTipaRada**, tabele **StavkaPodtipa** upisuje unetu vrednost kolone **nazivTipaRada** tabele **TipRadova** na osnovu vrednosti *sifraTipaRadova* |
| Update | nazivTipaRada | Da | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | Ne |  |

## Denormalizacija relacija uz narušavanje 3NF

Narušavanje 3NF na relaciji **Sredstva** **pre-joining** tehnikom denormalizacije:

**PRE DENORMALIZACIJE**



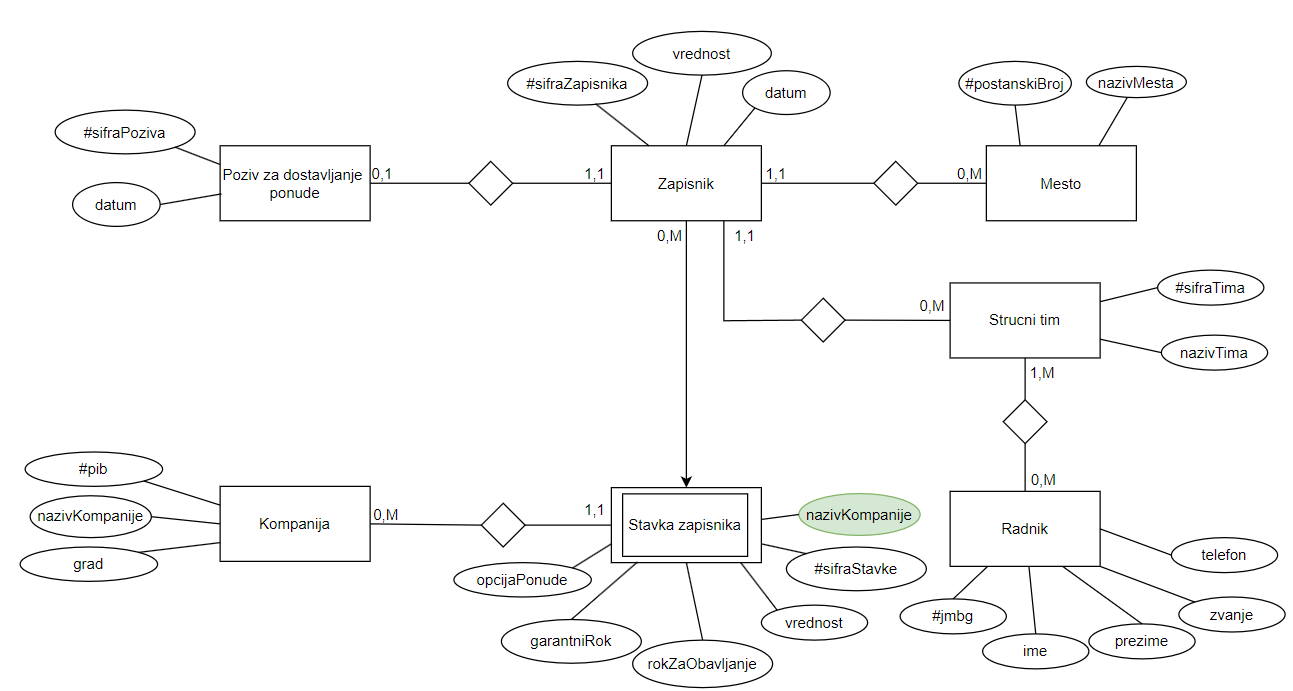
*Slika 25 - Zapisnik pre denormalizacije*

**StavkaZapisnika**(sifraZapisnika, sifraStavke, vrednost, rokZaObavljanje, garantniRok, opcijaPounde, *kompanijaId*)

**Kompanija**(kompanijaId, nazivKompanije, grad)

Funkcionalna zavisnost: sifraZapisnika, sifraStavke -> vrednost, rokZaObavljanje, garantniRok, opcijaPounde, kompanijaId

**NAKON DENORMALIZACIJE**



*Slika 36 - Zapisnik posle denormalizacije*

**StavkaZapisnika**(sifraZapisnika, sifraStavke, vrednost, rokZaObavljanje, garantniRok, opcijaPounde, nazivKompanije,  *kompanijaId*)

**Kompanija**(kompanijaId, nazivKompanije, grad)

Funkcionalna zavisnost: sifraZapisnika, sifraStavke -> vrednost, rokZaObavljanje, garantniRok, opcijaPounde, nazivKompanije, *kompanijaId*

kompanijaId -> nazivKompanije

**SPECIFIKACIJA TRIGERA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| Kompanija | Insert |  | Ne |  |
| Update | nazivKompanije | Da | Nakon izmene vrednosti kolone **nazivKompanije** u tabeli **Kompanija**, pokreće se triger koji izmenjenu vrednost ažurira i u tabeli **StavkaZapisnika**. |
| Delete |  | Ne |  |
| StavkaZapisnika | Insert |  | Da | U kolonu **nazivKompanije** tabele **StavkaZapisnika** upisuje unetu vrednost kolone **nazivKompanije** tabele **Kompanija** na osnovu vrednosti *kompanijaId* . |
| Update | kompanijaId | Da | Nakon izmene vrednosti kolone **kompanijaId**, pokreće se triger koji menja vrednost u koloni **nazivKompanije**. |
| nazivKompanije | Zabraniti direktno ažuriranje ove kolone. |
| Delete |  | Ne |  |

# KORISNIČKI DEFINISANI TIPOVI I TRIGERI

## Definicija korisničkom tipa i njegovo korišćenje

**DISTINCT TIP**

Definisanje DISTINCT korisnički definisanog tipa podatka *PIB\_TIP*  koji se definiše nad VARCHAR2 tipom podatka.

CREATE OR REPLACE TYPE PIB\_TIP AS OBJECT (

pib\_oznaka VARCHAR2(9),

MEMBER FUNCTION get\_pib\_tip RETURN VARCHAR2

) INSTANTIABLE FINAL;

CREATE OR REPLACE TYPE BODY PIB\_TIP AS

MEMBER FUNCTION get\_pib\_tip RETURN VARCHAR2 IS

BEGIN

RETURN SELF.pib\_oznaka;

END;

END;

Prikaz modifikacije postojećih relacija kroz naredbe kreiranja tabela i operacija nad tim relacijama, upotrebom DISTINCT korisnički definisanog tipa.

CREATE TABLE Kompanija (

kompanijaId NUMBER,

pib PIB\_TIP,

nazivKompanije VARCHAR2 (50),

grad VARCHAR2 (30),

constraint PK\_Kompanija PRIMARY KEY (kompanijaId),

check (REGEXP\_LIKE (PIB.pib\_oznaka,'[0-9]{9}'))

);

INSERT INTO Kompanija(kompanijaId,pib,nazivkompanije,grad) VALUES (1, PIB\_TIP('123456789'), 'VIRING', 'Beograd' );

INSERT INTO Kompanija(kompanijaId,pib,nazivkompanije,grad) VALUES (2, PIB\_TIP('126789345'), 'NIDZA +', 'Beograd' );

UPDATE Kompanija SET pib=pib\_tip('987654321') where kompanijaid = 1;

SELECT k.nazivkompanije,k.pib.get\_pib\_tip() FROM KOMPANIJA k;

SELECT \* FROM Kompanija;

**STRUKTUIRANI TIP**

Definisanje struktuiranog korisnički definisanog tipa podatka UKUPNA\_CENA\_TIP koji predstavlja ukupnu cenu na osnovu cene bez PDV-a i PDV-a.

CREATE OR REPLACE TYPE UKUPNA\_CENA\_TIP AS OBJECT (

vrednost DECIMAL(12,2),

pdv DECIMAL(3,2),

ukupna\_cena DECIMAL(12,2),

CONSTRUCTOR FUNCTION UKUPNA\_CENA\_TIP(vrednost DECIMAL, pdv DECIMAL) RETURN SELF AS RESULT,

MEMBER FUNCTION getCenaBezPdva RETURN DECIMAL,

MEMBER FUNCTION getPdv RETURN DECIMAL,

MEMBER FUNCTION getCenaSaPdvom RETURN DECIMAL

)INSTANTIABLE NOT FINAL;

CREATE OR REPLACE TYPE BODY UKUPNA\_CENA\_TIP AS

CONSTRUCTOR FUNCTION UKUPNA\_CENA\_TIP(u\_vrednost DECIMAL, u\_pdv DECIMAL) RETURN SELF AS RESULT IS

BEGIN

IF vrednost <0 or pdv<0 THEN

RAISE\_APPLICATION\_ERROR(-20000,'Vrednost cene i PDV ne mogu biti negativni brojevi'); END IF;

IF pdv>=1 THEN

RAISE\_APPLICATION\_ERROR(-20000,'PDV ne moze biti veci od 100% cene bez PDV-a');

END IF;

SELF.vrednost:=vrednost;

SELF.pdv:=pdv;

SELF.ukupna\_cena:= vrednost+ vrednost\*pdv;

RETURN; END;

MEMBER FUNCTION getCenaBezPdva RETURN DECIMAL IS

BEGIN

RETURN SELF.vrednost;

END;

MEMBER FUNCTION getPdv RETURN DECIMAL IS

BEGIN

RETURN SELF.pdv;

END;

MEMBER FUNCTION getCenaSaPdvom RETURN DECIMAL IS

BEGIN

RETURN SELF.ukupna\_cena;

END;

END;

CREATE TABLE Narudzbenica (

narudzbenicaId NUMBER,

cena UKUPNA\_CENA\_TIP,

rokIzvrsenja DATE,

garantniRok DATE,

nacinPlacanja VARCHAR2(50),

datum DATE,

postanskiBroj NUMBER,

zapisnikId NUMBER,

nacelnikId NUMBER,

kompanijaId NUMBER,

constraint PK\_Narudzbenica PRIMARY KEY (narudzbenicaId),

constraint FK\_Narudzbenica FOREIGN KEY (postanskiBroj) REFERENCES Mesto(postanskiBroj),

FOREIGN KEY (zapisnikId) REFERENCES Zapisnik(zapisnikId),

FOREIGN KEY (nacelnikId) REFERENCES Nacelnik(nacelnikId),

FOREIGN KEY (kompanijaId) REFERENCES Kompanija(kompanijaId)

);

INSERT INTO Narudzbenica VALUES(1,ukupna\_cena\_tip(1000000,0.15),TO\_DATE('23/12/2023','dd/mm/yyyy'),TO\_DATE('23/12/2024','dd/mm/yyyy'),'3 rate', TO\_DATE('30/11/2023','dd/mm/yyyy'),11000,1,1,1);

INSERT INTO Narudzbenica VALUES(2,ukupna\_cena\_tip(999.99,0.20),TO\_DATE('23/12/2023','dd/mm/yyyy'),TO\_DATE('23/12/2024','dd/mm/yyyy'),'3 rate', TO\_DATE('30/11/2023','dd/mm/yyyy'),11000,1,1,1);

INSERT INTO Narudzbenica VALUES(3,ukupna\_cena\_tip(999.99,0.8),TO\_DATE('23/12/2023','dd/mm/yyyy'),TO\_DATE('23/12/2024','dd/mm/yyyy'),'3 rate', TO\_DATE('30/11/2023','dd/mm/yyyy'),11000,1,1,1);

SELECT n.narudzbenicaId,n.cena.getCenaSaPdvom(),n.cena.getCenaBezPdva(),n.cena.getPdv(),n.datum FROM Narudzbenica n;

SELECT \* FROM Narudzbenica;

## Realizacija proceduralne logike uz pomoć trigera

Implementacija trigera definisanih u specifikaciji trigera prilikom narušavanja **2NF** (pre-joining keys):

1. Triger koji prilikom izmene vrednosti kolone *nazivTipaRada* u tabeli **TipRadova**, postavlja

vrednost kolone **nazivTipaRada** odgovarajucim rekordima tabele **StavkaPodtipa.**

CREATE OR REPLACE TRIGGER TR\_TIPRADOVA\_UPDATE\_NAZIV\_TIPA\_RADA

BEFORE UPDATE OF nazivTipaRada ON TipRadova

FOR EACH ROW

DECLARE

v\_noviNaziv VARCHAR2(50);

v\_tipRadovaId NUMBER;

BEGIN

v\_noviNaziv := :NEW.nazivTipaRada;

v\_tipRadovaId := :NEW.tipRadovaId;

DBMS\_OUTPUT.PUT\_LINE('TR\_TIPRADOVA\_UPDATE\_NAZIV\_TIPA\_RADA is executing. tipRadovaId: ' || v\_tipRadovaId);

UPDATE StavkaPodtipaRadova

SET nazivTipaRada = v\_noviNaziv

WHERE tipRadovaId = v\_tipRadovaId;

-- Nastavite sa izmenama samo ako je dozvoljeno

END;

1. Triger koji postavlja vrednost kolone *nazivTipaRada*, tabele **StavkaPodtipa** upisuje unetu vrednost kolone **nazivTipaRada** tabele **TipRadova** na osnovu vrednosti *SifraTipaRadova*

CREATE OR REPLACE TRIGGER TR\_STAVKAPODTIPARADOVA\_INSERT\_NAZIV\_TIPA\_RADA

BEFORE INSERT

ON StavkaPodtipaRadova

FOR EACH ROW

DECLARE

v\_nazivTipaRada varchar2 (50);

BEGIN

SELECT tp.nazivTipaRada

INTO v\_nazivTipaRada

FROM TipRadova tp

WHERE tp.tipRadovaId = :NEW.tipRadovaId;

:NEW.nazivTipaRada:= v\_nazivTipaRada;

END;

1. Triger koji sprečava direktnu izmenu kolone *nazibTipaRada* tabele StavkaPodtipa

CREATE OR REPLACE TRIGGER TR\_STAVKAPODTIPARADOVA\_FORBID\_UPDATE\_NAZIV\_TIPA\_RADA

BEFORE UPDATE OF nazivTipaRada ON StavkaPodtipaRadova

FOR EACH ROW

DECLARE

v\_callStack VARCHAR2(4000); -- Pretpostavljena maksimalna dužina call stack-a

v\_allowUpdate NUMBER;

BEGIN

v\_callStack := DBMS\_UTILITY.FORMAT\_CALL\_STACK;

DBMS\_OUTPUT.PUT\_LINE('TR\_STAVKAPODTIPARADOVA\_FORBID\_UPDATE\_NAZIV\_TIPA\_RADA is executing. Call Stack: ' || v\_callStack);

-- Proverite da li je drugi trigger u call stack-u

IF INSTR(v\_callStack, 'TR\_TIPRADOVA\_UPDATE\_NAZIV\_TIPA\_RADA') > 0 THEN

-- Trigger zabrane je aktiviran od strane drugog triggera, dozvolite izmenu

v\_allowUpdate := 1;

ELSE

-- Drugi trigger nije u call stack-u, zabranite izmenu

v\_allowUpdate := 0;

END IF;

-- Proverite da li je dozvoljena izmena

IF v\_allowUpdate = 1 THEN

NULL;

ELSE

RAISE\_APPLICATION\_ERROR (-20000, 'Zabranjeno azuriranje kolone nazivTipaRada u okviru tabele StavkaPodtipaRadova!!!');

END IF;

END;

Implementacija trigera definisanih u specifikaciji trigera prilikom narušavanja **3NF** (pre-joining keys):

1. Triger koji prilikom promene vrednosti kolone ***nazivKompanije*** tabele **Kompanija**, postavlja vrednost kolone **nazivKompanije** odgovarajućim rekordima tabele StavkaZapisnika na osnovu spoljnog ključa **kompanijaId.**

CREATE OR REPLACE TRIGGER TR\_KOMPANIJA\_UPDATE

AFTER UPDATE OF nazivKompanije ON Kompanija

FOR EACH ROW

DECLARE

v\_noviNaziv VARCHAR2(50);

v\_kompanijaid NUMBER;

BEGIN

v\_noviNaziv := :NEW.nazivKompanije;

v\_kompanijaid := :NEW.kompanijaid;

DBMS\_OUTPUT.PUT\_LINE('TR\_KOMPANIJA\_UPDATE is executing. kompanijaid: ' || v\_kompanijaid);

UPDATE StavkaZapisnika

SET nazivKompanije = v\_noviNaziv

WHERE kompanijaid = v\_kompanijaid;

END;

1. Triger koji postavlja vrednost kolone *nazivKompanije* u tabeli **StavkaZapisnika** na osnovu vrednosti **kompanijaId**

CREATE OR REPLACE TRIGGER TR\_STAVKAZAPISNIKA\_INSERT

BEFORE INSERT ON StavkaZapisnika

FOR EACH ROW

DECLARE

v\_nazivKompanije VARCHAR2(50);

BEGIN

SELECT nazivKompanije INTO v\_nazivKompanije

FROM Kompanija

WHERE kompanijaId = :NEW.kompanijaId;

:NEW.nazivKompanije:=v\_nazivKompanije;

END;

1. Triger koji nakon izmene vrednosti kolone **kompanijaId**, pokreće se triger koji menja vrednost u koloni **NazivKompanije**.

CREATE OR REPLACE TRIGGER TR\_STAVKAZAPISNIKA\_UPDATE\_KOMPANIJAID

BEFORE UPDATE OF kompanijaId ON StavkaZapisnika

FOR EACH ROW

DECLARE

v\_nazivKompanije VARCHAR2(30);

BEGIN

SELECT k.nazivKompanije INTO v\_nazivKompanije

FROM Kompanija k

WHERE k.kompanijaid = :NEW.kompanijaId;

:NEW.nazivKompanije := v\_nazivKompanije;

END;

1. Triger koji sprečava direktnu izmenu kolone *nazivKompanije* tabele **StavkaZapisnika**

CREATE OR REPLACE TRIGGER TR\_FORBID\_KOMPANIJA\_UPDATE

BEFORE UPDATE OF nazivKompanije ON StavkaZapisnika

FOR EACH ROW

DECLARE

v\_callStack VARCHAR2(4000); -- Pretpostavljena maksimalna dužina call stack-a

v\_allowUpdate NUMBER;

BEGIN

v\_callStack := DBMS\_UTILITY.FORMAT\_CALL\_STACK;

DBMS\_OUTPUT.PUT\_LINE('TR\_FORBID\_KOMPANIJA\_UPDATE is executing. Call Stack: ' || v\_callStack);

-- Proverite da li je drugi trigger u call stack-u

IF INSTR(v\_callStack, 'TR\_KOMPANIJA\_UPDATE') > 0 THEN

-- Trigger zabrane je aktiviran od strane drugog triggera, dozvolite izmenu

v\_allowUpdate := 1;

ELSE

-- Drugi trigger nije u call stack-u, zabranite izmenu

v\_allowUpdate := 0;

END IF;

-- Proverite da li je dozvoljena izmena

IF v\_allowUpdate = 1 THEN

NULL;

ELSE

RAISE\_APPLICATION\_ERROR (-20000, 'Nije dozvoljena direktna izmena naziva kompanije u okviru tabele StavkaZapisnika!!!');

END IF;

END;

# OPTIMIZACIJA BAZE PODATAKA

## Definisanje i analiza indeksa

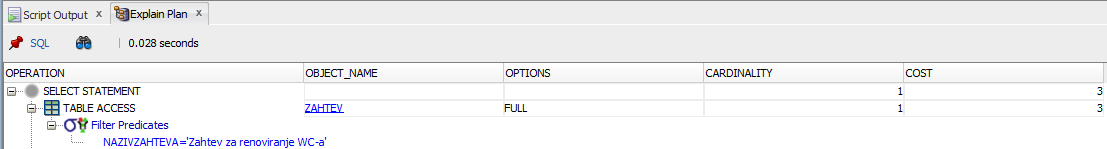
**Index nad tekstualnim poljem**

Ovaj index će biti kreiranu usled česte potrebe za traženjem određenog zahteva na osnovu njegovog imena.

Upit:



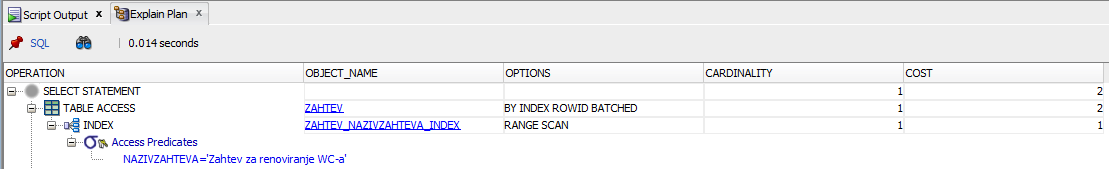
Izvršenja upita pre uvođenja indeksa ZAHTEV\_NAZIVZAHTEVA\_INDEX:



Definisanje index-a:



Izvršenja upita nakon uvođenja indeksa ZAHTEV\_NAZIVZAHTEVA\_INDEX:



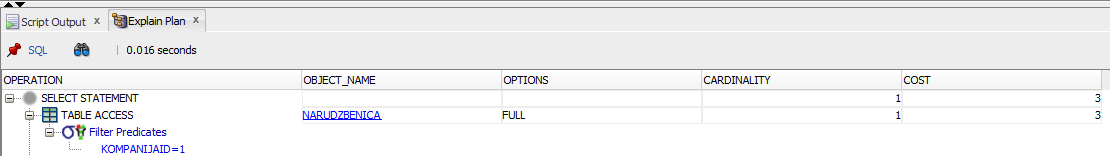
**Index nad spoljnim ključem**

Ovaj index će biti kreiranu usled česte potrebe za traženjem narudžbenoca sa određenim kompanijama preko njihovog id-a.

Upit:



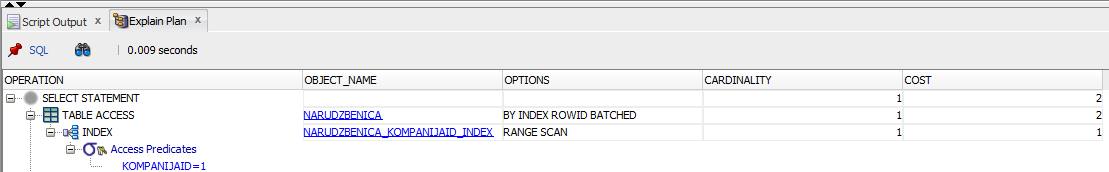
Izvršenja upita pre uvođenja indeksa NARUDZBENICA\_KOMPANIJAID\_INDEX:



Definisanje index-a:

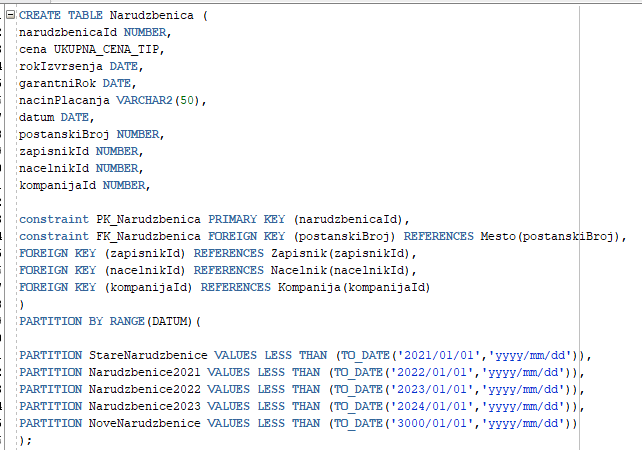


Izvršenja upita nakon uvođenja indeksa NARUDZBENICA\_KOMPANIJAID\_INDEX:



## Horizontalno particionisanje

Usled česte pretrage narudžbenica u prethodnih 3 godine napravljeno je horizontalno particionisanje na osnovu datum kad je narudžbenica kreirana.



Primer upotrebe particija:



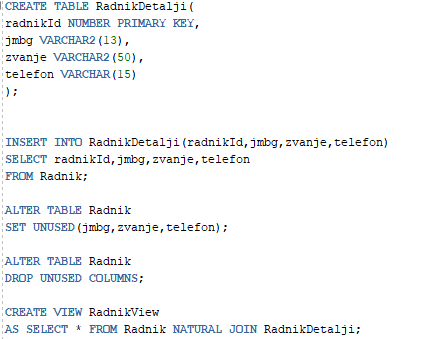
## Vertikalno particionisanje

Za potrebe ovog primera biće izvršano vertikalno particionisanje tabele **Radnik** na podatke koji se češće koriste i one koji se smatraju detaljnijim informacijama kojima se ređe pristupa. Prvo je kreirana tabela **RadnikDetalji** sa odgovarajućim kolonama te je sadržaj odgovarajućih kolona iz tabele Radnik upisan u novu kreiranu tabelu. Zatim su reduntante kolone izbačene iz inicijalne tabele. Naposletku, kreiran je **pogled** kako bi apstrahovao rad sa particijama uz odgovarajući triger.

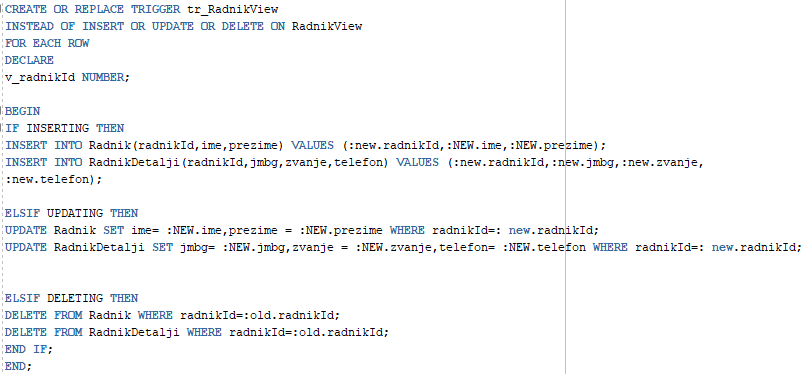
Relacioni model:

Radnik(radnikId*,*ime,prezime),

StavkaZapisnikaDetalji(radnikId,jmbg,zvanje,telefon)



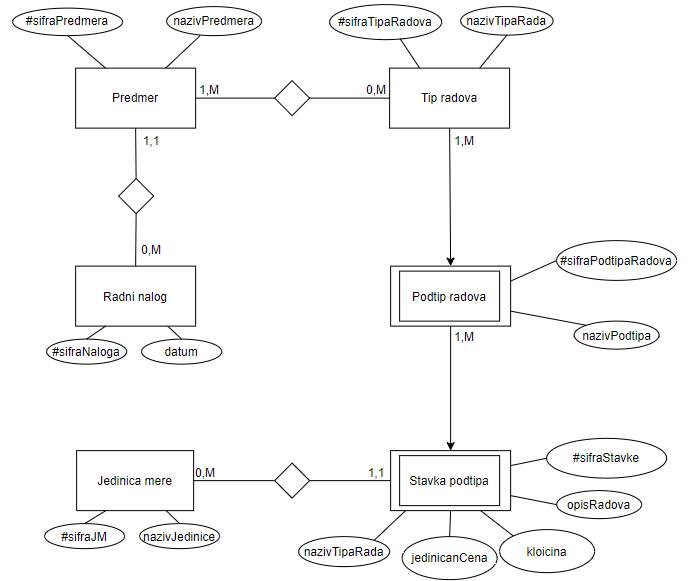
Kreiranje **INSTEAD OF** trigera nad pogledom za odgovarajuće **DML** naredbe:



## Primena drugih optimizacionih tehnika

**Storing Derivable Values**

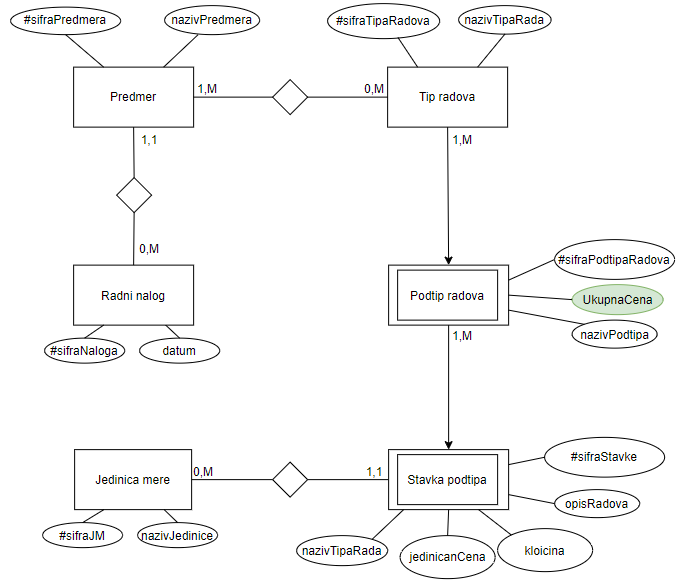
Konceptualni model pre optimizacije:



PodtipRadova(*sifraTipaRadova*,sifraPodtipaRadova*,* nazivTipaRada)

Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCenA, nazivTipaRada, *sifraJM*)

Konceptualni model posle optimizacije:



PodtipRadova(*sifraTipaRadova*,sifraPodtipaRadova*,* nazivTipaRada, ukupnaCena)

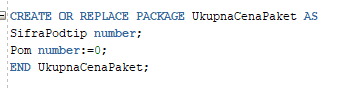
Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCena, nazivTipaRada, *sifraJM*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tip trigera** | **Kolona** | **Potreban** | **Šta treba da uradi?** |
| PodtipRadova | Insert |  | Ne |  |
| Update | ukupnaCena | Da | Zabrana direktne izmene polja **ukupnaCena.** |
| Delete |  | Ne |  |
| StavkaPodtipa | Insert |  | Da | Prilikom unosa nove **StavkaPodtipa** , okida se triger koji poziva proceduru za ažuriranje  **ukupnaCena** na PodtipRadova |
| Update | kolicina | Da | Prilikom ažuriranja **kolicina** **StavkaPodtipa** , okida se triger koji poziva proceduru za ažuriranje  **ukupnaCena** na PodtipRadova |
| jedinicnaCena | Prilikom ažuriranja **jedinicnaCena StavkaPodtipa** , okida se triger koji poziva proceduru za ažuriranje  **ukupnaCena** na PodtipRadova |
| Delete |  | Da | Prilikom brisanja **StavkaPodtipa** , okida se triger koji poziva proceduru za ažuriranje  **ukupnaCena** na PodtipRadova |

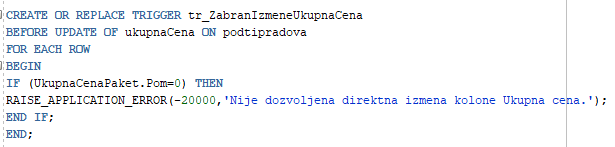
Uvođenje nove kolone u kojoj će se računati ukupna cena za **PodtipRadova**.



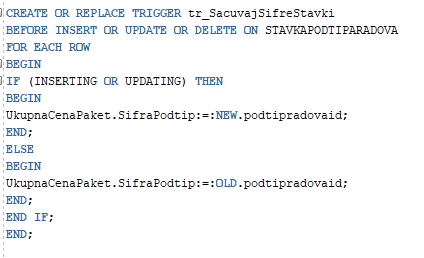
Uvođenje paketa, koji sadrži indikator, kako bi se razrešio konflikt između trigera:



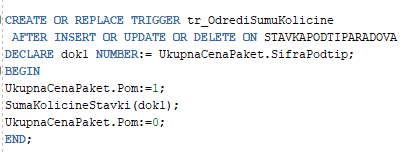
Triger koji sprečava direktnu izmenu kolone **ukupnaCena**:



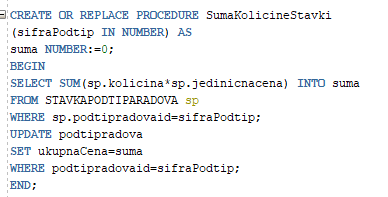
Triger pomoću kog se određuje šifra I postavlja u globalnu promenljivu:



Triger koji poziva proceduru:

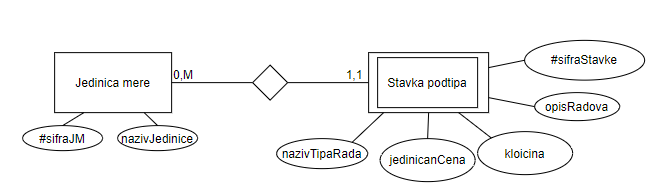


Procedura za određivanje ukupne cene:



**Hard-coded values**

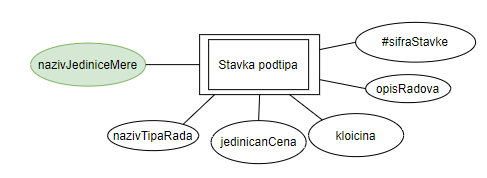
Model pre optimizacije:



Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCena, *sifraJM*)

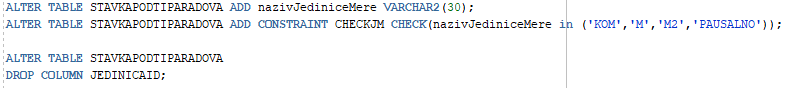
JedinicaMere(sifraJM, nazivJedinice)

Model posle optimizacije:



Stavkapodtipa(*sifraTipaRadova*, *sifraPodtipaRadova*, sifraStavke, opisRadova, kolicina, jedinicnaCena nazivJediniceMere)

Constraint koji se dodaje zbog hard-coded values tehnike optimizacije i naredbe koje se izvršavaju:



# TEHNOLOGIJE ZA IMPLEMENTACIJU PROJEKTA

## SUBP korišćen za implementaciju baze podatka (kratak opis)

## U procesu implementacije baze podataka, Oracle je odabran kao ključni sistem za upravljanje podacima. Oracle se ističe zbog svoje izuzetne pouzdanosti, skalabilnosti i visokih performansi, čineći ga optimalnim izborom za zadovoljenje zahteva složenih poslovnih potreba. Korišćenje Oracle-a omogućava efikasno upravljanje podacima uz podršku za napredne funkcionalnosti, visoke sigurnosne standarde i sposobnost rukovanja kompleksnim transakcijama. Uz to, za razvoj i održavanje baze podataka koristi se SQL Developer, alat koji dodatno olakšava rad s Oracle bazom podataka putem intuitivnog korisničkog interfejsa i optimizacije SQL upita.

## Programsko okruženje za razvoj korisničkog interfejsa ( kratak opis)

## Pri razvoju korisničkog interfejsa, korišćeno je programsko okruženje zasnovano na programskom jeziku Java. Aplikacija je implementirana u NetBeans15 razvojnom okruženju, uz upotrebu Swing biblioteke za kreiranje korisničkog interfejsa. Java je odabrana zbog svoje svestranosti i moćnih mogućnosti programiranja, pružajući stabilnost i performanse. NetBeans, kao integrirano razvojno okruženje, olakšava procese razvoja aplikacija. Kroz primenu Swing biblioteke postignuta je jednostavna izrada korisničkog interfejsa, prilagodljivog potrebama aplikacije, čime se korisnicima pruža intuitivno i efikasno iskustvo korišćenja.